Sven Schmidt, Rainer Gemulla, Wolfgang Lehner

XML Stream Processing Quality

Erstveröffentlichung in / First published in:

Database and XML Technologies. First International XML Database Symposium, XSYM 2003. Berlin, 08.09.2003. Springer, S. 195-207. ISBN 978-3-540-39429-7.

DOI: http://dx.doi.org/10.1007/978-3-540-39429-7_13

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

XML Stream Processing Quality

Sven Schmidt, Rainer Gemulla, and Wolfgang Lehner

Dresden University of Technology, Germany
{ss54,rg654452,lehner}@inf.tu-dresden.de,
http://www.db.inf.tu-dresden.de

Abstract. Systems for selective dissemination of information (SDI) are used to efficiently filter, transform, and route incoming XML documents according to pre-registered XPath profiles to subscribers. Recent work focuses on the efficient implementation of the SDI core/filtering engine. Surprisingly, all systems are based on the best effort principle: The resulting XML document is delivered to the consumer as soon as the filtering engine has successfully finished. In this paper, we argue that a more specific Quality-of-Service consideration has to be applied to this scenario. We give a comprehensive motivation of quality of service in SDI-systems, discuss the two most critical factors of XML document size and shape and XPath structure and length, and finally outline our current prototype of a Quality-of-Service-based SDI-system implementation based on a real-time operating system and an extention of the XML toolkit.

1 Introduction

XML documents reflect the state-of-the-art for the exchange of electronic documents. The simplicity of the document structure in combination with comprehensive schema support are the main reason for this success story. A special kind of document exchange is performed in XML-based SDI systems (selective dissemination systems) following the publish/subscribe communication pattern between an information producer and information subscriber. On the one hand, XML documents are generated by a huge number and heterogeneous set of publishing components (publisher) and given to a (at least logically) central message broker. On the other hand, information consumers (subscriber) are registering subscriptions at the message broker usually using XPath or XQuery/XSLT expressions to denote the profile and delivery constraints. The message broker has to process incoming by filtering (in the case of XPath) or transforming (in the case of XQuery/XSLT) the original documents and deliver the result to the subscriber (figure 1).

Processing XML documents within this streaming XML document application is usually done on a best effort basis i.e. subscribers are allowed to specify only functionally oriented parameters within their profiles (like filtering expressions) but no parameters addressing the quality of the SDI service. Quality-of-Service in the context of XML-based SDI systems is absolutely necessary for example in application area of stock exchange, where trade-or-move messages

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

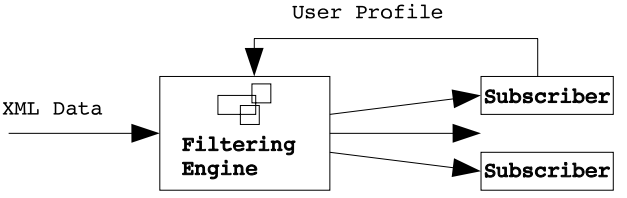

Fig. 1. basic logical architecture of SDI systems

have to be delivered to registered users within a specific time slot so that given deadlines can be met. Although a quality-of-service-based process scheduling of XML filtering operations yields typically less overall system throughput, the negotiated quality-of-service for the users can be guaranteed.

Contribution of the Paper: Scheduling and capacity planning in the context of XML documents and XPath expression evaluation is difficult but may be achieved within a certain framework. This topic is intensively discussed in the context of this paper. Specifically, the paper illustrates how the resource consumption of filtering, a typical operation in SDI systems, depends on the shape, size and complexity of the document, on the user profile specified as a filter expression, and on the efficiency of the processor which runs the filters against the documents. We finally sketch an XML-based SDI system environment which is based on a real-time operating system and is thus capable of providing Quality-of-Service for subscribers.

Structure of the Paper: The paper is organized as follows: In the next section, the current work in the area of XML-processing related to our approach is summarized. Section 3 considers Quality-of-Service perspectives for data processing in SDI systems and proposes a list of requirements regarding the predictability of XML data, filters and processors to consequently guarantee a user-defined quality of service. In section 4 the QoS parameters are used to obtain resource limits for QoS planning and in section 5 ideas about the architecture of a QoS-capable SDI system are given. Section 6 outlines the current state of our prototypical implementation based on the XML toolkit and on a real-time operating system. Section 7 finally concludes the paper with a short summary.

2 Related Work

The process of efficiently filtering and analyzing streaming data is intensively discussed in recent publications. Many mechanisms to evaluate continuous/standing queries against XML documents have been published. The work in this area ranges from pure processing efficiency to the handling of different data sources

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

[1], adoption of the query process by dynamic routing of tuples [5] and grouping of queries based on similarity including dynamic optimization of these query groups [12]. Surprisingly and to the best of our knowledge, no system incorporates the idea of quality of service for the filtering process in SDI systems as a first-class parameter. Since our techniques and parameter are based on previous work, we have to sketch the accompanying techniques:

One way to establish the filtering of XML documents with XPath expressions consists in using the standard DOM representation of the document. Unfortunately, using the DOM representation is not feasible for larger XML documents. The alternative way consists in relying on XML stream processing techniques [2,8,4,3] which particularly construct automatons based on the filter expressions or use special indexes on the streaming data. This class of XPath evaluations will be the base for our prototypical implementation outlined in section 6.

In [13] some basic XML metrics are used to characterize the document structure. Although their application area is completely different to Quality-of-Service in XML-based SDI systems, we exploit the idea of XML metrics as a base to estimate the resource consumption for the filtering process of a particular XML document.

3 XML–Based QoS–Perspectives

Before diving into detail, we have to outline the term "Quality-of-Service" in the context of SDI systems. In general a user is encouraged to specify QoS requirements regarding a job or a process a certain system has to perform. These requirements usually reflect the result of a negotiation between user and system. Once the system has accepted the user's QoS requirement, the system guarantees to meet these requirements. Simple examples of quality subjects are a certain precision of a result or meeting a deadline while performing the user's task.

The benefit for the user is predictability regarding the quality of the result or the maximal delay of receiving the result. This is helpful in a way that users are able to plan ahead other jobs in conjunction with the first one. As a consequence from the system perspective, adequate policies for handling QoS constraints have to be implemented. For example to guarantee that a job is able to consume a certain amount of memory during its execution, all the memory reservations have to be done in advance when assuring the quality (in this case the amount of available memory). In most cases even the deadline of the job execution is specified as a quality of service constraint. A job is known to require a certain amount of time or an amount of CPU slices to finish. Depending on concurrently running jobs in the system a specific resource manager is responsible for allocating the available CPU slices depending on the QoS specified time constraints. Most interesting from an SDI point of view is that every time a new job negotiates about available computation time or resources in general, an admission control has to either accept or reject the job according to the QoS requirements.

QoS management is well known for multimedia systems especially when dealing with time dependent media objects like audio and video streams. In such a

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

case the compliance to QoS requirements may result in video playback without dropping frames or in recording audio streams with an ensured sampling frequency.

Depending on the type of SDI system, deadlines in execution time or in data transmission are required from a user point of view. An example is the NASDAQ requirement regarding the response time to Trade-or-Move messages or (more generally) the message throughput in the stock exchange systems like Philadelphia Stock Exchange which are measured in nearly one hundred thousand messages (and therefore filtering processes) per second. To ensure quality of service for each single SDI subscriber job and fairness between all subscribers, SDI systems based on the best effort principle (i.e. process incoming messages as fast as the can without any further optimization and scheduling) are not sufficient for those critical applications. A solid basis should be systems with a guaranteed quality of its services.

Figure 2 shows the components which have to be considered when discussing quality of service in the context of XML-based SDI systems. The data part consists of XML messages which stream into the system. They are filtered by a XPath processor operating on top of a QoS capable operating system.

- processor: the algorithm of the filtering processor has to be evaluated with regard to predictability. This implies that non-deterministic algorithms can be considered only on a probability basis, while the runtime of deterministic algorithms can be precomputed for a given set of parameters.
- data: the shape and size of an XML document is one critical factor to determine the behavior of the algorithm. In our approach, we exploit metrics (special statistics) of individual XML documents to estimate the required capacity for the filtering process in order to meet the quality of service constraints.
- query: the second determining factor is the size and structure of the query to filter (in the case of XPath) or to transform (in the case of XQuery/XSLT) the incoming XML document. In our approach, we refer to the type and number of different location steps of XPath expressions denoting valid and individual subscriptions.
- QoS capable environment: the most critical point in building an SDI system considering QoS parameters is the existence of an adequate environment. As shown in section 6.1, we rely on a state-of-the-art real-time operating system which provides native streaming support with QoS. Ordinary best effort operating systems are usually not able to guarantee a certain amount of CPU time and/or data transfer rate to meet the subscription requirement.

4 Using Statistics

As outlined above, the shape and size of an XML document as well as the length and structure of the XPath expressions are the most critical factors estimating the overall resource consumption regarding a specific filter algorithm. The factors are described in the remainder of this section.

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

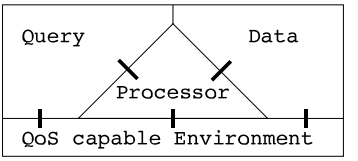

Fig. 2. QoS determining factors in XML-based subscription systems

4.1 Complexity of XML Data

In [13] different parameters for describing the structure of XML documents and schemes are outlined on a very abstract level. The so called *metrics* evolve from certain scheme properties and are based on the graphical representation of the XML scheme. The five identified metrics are:

- size: counted elements and attributes
- structure: number of recursions, IDREFs
- tree depth
- fan-in: number edges which leave a node
- fan-out: number of edges which point to a node

Obviously these metrics are related to the complexity of a document and strongly influence the resources needed to query these data. Depending on the requirements of the specific SDI system we may add some more parameters or we may only record metrics on a higher level (like the documents DTD). However, the question is how to obtain these statistics. We propose three different directions, outlined in the following:

- producer given statistics: We require the document statistics from the producer of an XML document. The statistics are gathered during the production process of the document and transferred to the filtering engine together with informational payload. This method, however, requires cooperative producers, fulfilling the requirements prescribed in a producer-filtering engine document transmission protocol. Examples are parameters like the DTD of a document (or of a collection of documents) and the document size (length) itself.
- generating statistics: We apply the method of gathering statistics in centralized systems to the SDI environment. This approach however implies that the stream of data will be broken because the incoming data has to be preprocessed and therefore stored temporarily. As soon as the preprocessing has completely finished, the actual filtering process may be initiated. Obviously, this pipeline breaking behavior of the naive statistic gathering method does not reflect a sound basis for efficiently operating SDI systems.

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

– cumulative statistics: as an alternative to the producer given statistics we start with default values. Then statistics of the first document are gathered during the filtering step. These statistical values are merged with the default values and used to estimate the overhead for the following document of the same producer. In general, the statistics of the i-th document are merged with the statistics of the documents 1 to i-1 and used to perform the capacity planning of the i+1-th document of the same producer. This method can be applied only in a "static" producer environment.

The assumption of creating data statistics at the data source is relatively strong but might improve the overall quality of the system tremendously. As a result of the above discussion the, set of document statistics to be used for QoS planning has to be chosen carefully in terms of resource consumption of the filtering engine. Section 5 gives further explanations on this.

4.2 Complexity of XPath Evaluation

In the context of XPath evaluation, the structure and the number of the XPath expressions are combined with the filter algorithm itself. It does not make sense to consider these two perspectives (i.e. XPath and processor) independently from each other because the requirements regarding the XPath expressions strongly vary in terms of the underlying evaluation engine.

Due to extensive main memory requirements, the well-known DOM based evaluation is not applicable for the purpose of SDI systems and will not be considered in this paper. Therefore we focus on the family of stream based XML filtering algorithms. One of the main ideas in this field is using an automaton which is constructed with regard to the set of given XPath expressions reflecting single subscriptions. Such an automaton has a number of different states which may become active while the processor is scanning through the XML document. The set of techniques may be classified according to the deterministic or non-deterministic behavior of the automaton.

Whereas for an NFA (non-deterministic finite automaton) the required amount of memory for representing the states determined by the number of states per automaton and the number of XPath expressions is known in advance, the processing time is indeterministic and difficult to estimate. In opposite to the NFA, the DFA (deterministic finite automaton) has no indeterminism regarding the state transitions but consumes more memory because of the amount of potential existing automaton states. In real application scenarios with thousands of registered XPath expressions it is not possible to construct all automaton states in main memory. The solution provided in [3] is to construct a state when it is needed the first time (data driven).

From the QoS point of view, XPath evaluation mechanisms with predictable resource consumption are of interest. It is necessary to consider worst-case and best-case scenarios as the basic resource limits. In the case of DFA worst case assumptions will not be sufficient, because the worst case is constructing *all*

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

states regardless of the XML document, so that more accurate approaches for estimating memory and CPU usage are required.

We make use of the XML toolkit implementation as a representative of deterministic automatons. For gathering the QoS parameters basically the memory consumption and the CPU usage are considered. In the XMLTK a lazy DFA is implemented. This means that a state is constructed on demand so that memory requirements may be reduced.

[3] proposes different methods for gathering the resource requirements of XML toolkit automaton. This is possible when making some restrictions regarding the data to be processed and regarding the filtering expressions. For example the XML documents have to follow a simple DTD[1] and the XPath expressions have to be linear and may only make use of a small set of location steps.

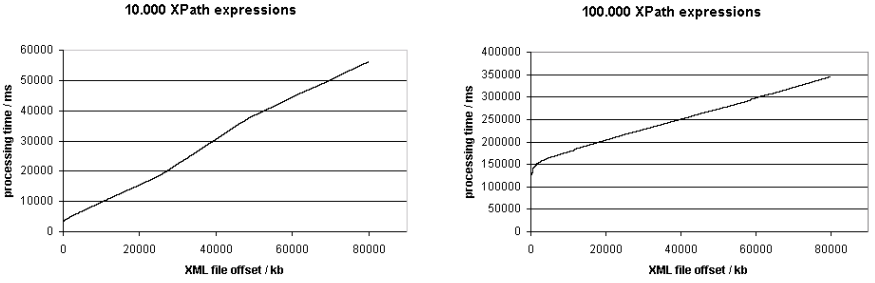

Fig. 3. CPU Usage with increasing number of XPath expressions

CPU Resource Utilization: Fortunately, one property of a lazy DFA is less overall memory consumption. The drawback are delays for state transitions in the warm-up phase. The cumulative time of state transitions and state creations is illustrated in figure 3. As long as not all states are constructed, the time needed for one state transition consists of the state creation time and the transition time itself. In terms of resource management the following approach may help: The time required for a certain number of state transitions may be calculated as follows:

$t(x) \leq x * t_s + t_{c-all}$

where x is the number of the steps initiating a state transition[2], t_s is the time required for a state transition (assumed to be constant for one automaton, independently of the number of registered XPath expressions) and t_{c-all} is the

[1] No cycles are allowed except to the own node.

[2] Every open and close tag causes a state transition. Therefore it should be possible to use statistics for estimating the number of state transitions regarding the XML file size.

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

time required for the creation of *all* states of the automaton (the time required for creating *one* state depends on the number of registered XPath expressions, so we use the cumulative value here).

The number of constructed states in the warm-up phase is obviously smaller then the number of all states required by the document. Using the time required to construct *all* states (t_{c-all}) in the formula will give an upper bound of the computation time. Assuming the warm-up phase to be shorter than the rest of the operating time, $t(x)$ is a reasonable parameter for resource planning.

Using the sketched approach of CPU utilization, the time for processing a single document may be scheduled just before the document arrives at the DFA. In section 5 an architecture for filtering subsequent XML documents is proposed.

Memory Consumption: Regarding to [3] there is an upper bound of the number of constructed states for a lazy DFA. This value depends on the structure of the registered XPath expressions as well as on the DTD of the XML documents. We assume that the memory requirements for each state can be calculated, so this upper bound may also be used for estimating an overall memory consumption better then worst-case. Having the estimated number of states and the memory used per state available, the required memory can be calculated. Hence for a static set of registered filter expressions and for a set of documents following *one* DTD the required memory is known and may be reserved in advance.

5 Architecture of a QoS-Capable SDI System

In SDI systems it is common that users subscribe to receive a certain kind of information they are interested and a (static) set of data sources register their services at the SDI system with a certain information profile.

This results in consecutive XML documents related to each other. These relationships may be used to optimize the statistic runs. Consider an example like stock exchange information received periodically from a registered data source (from a stock exchange). The consecutive documents reflect update operations in the sense that an update may exhibit the whole stock exchange document with partially modified exchange rates or it *only* consists of the updated exchange rates wrapped by an XML document. In summary, updates logically consist of:

- element content update
- update in document structure
- updated document with a new DTD or new XML scheme

All three kinds of update have to be accepted to preserve the flexibility of XML as a data exchange format.

Figure 4 sketches the idea of a QoS-capable SDI system. Starting with one data source disseminating a sequence of XML documents, some consecutive documents will follow the same DTD. The DTD is taken as a basis for the first set of QoS-determining parameters (I). On the subscriber side many XPath expressions are registered at the SDI system. The structure and length of these subscriptions

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

forms the second parameter set (II). Finally the XML document characteristics (in our case only the document length) is the third parameter (III). All parameters are used by the scheduler which is able to determine low-level resources like main memory and CPU consumption accordingly to the deployed filter mechanism. After negotiating with the operating systems resource manager, a new document filter job may be admitted or rejected (admission control). In the former case the scheduler reserves the resources at the resource manager and - as a consequence - the real-time operating system environment has to guarantee the required memory as well as the CPU time. The filtering job may then start and will successful finish after the predetermined amount of time.

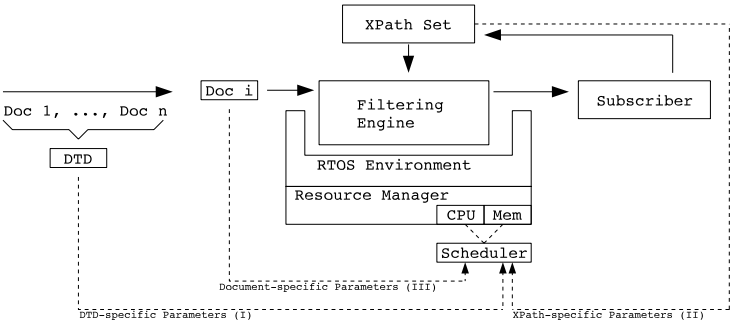

Fig. 4. data and information flow in a QoS SDI system

Periods: As soon as one of the factors (I, II, III) changes, the filter process must be re-scheduled. Due to the nature of an SDI system we assume the set of XPath expressions to be fixed for a long time (continuous/standing queries). The data will arrive at the SDI system depending on the frequency of dissemination of the data sources, so the shortest period will be the processing time of one single document followed by the period the DTD changes[3].

Unfortunately it is hard to create only *one* schedule for *a sequence* of documents because the documents parameter (III, the documents length) seems unpredictable in our case. As a result the sketched SDI system will initially perform an ad-hoc scheduling for each arriving XML document independently. A periodic behavior of the filtering system may be realized on a macro level (e.g. in cases of document updates while not changing the document structure and size) or on a micro level (a certain amount of CPU time is reserved for performing the different state transitions).

[3] In a worst case every new XML document follows another DTD. Hopefully lots of documents of the same data source will follow the same DTD.

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

6 Implementational Perspectives

As already outline in section 3 the use of a real-time operating system (RTOS) reflects a necessary precondition for the stated purpose of pushing QoS into SDI systems.

6.1 Real-Time Operating System Basis

A common property of existing RTOSes is the ability to reserve and to assure resources for user processes. Generally there are the two types of the soft and hard real-time systems. In hard real-time systems the resources, once assured to a process, have to be realized without any compromise. Examples are systems for controlling peripheral devices in critical application environments such as in medicine. Real-time multimedia systems for example are classified as soft real-time systems, because it is tolerable to drop a single frame during a video playback or to jitter in sampling frequency while recording an audio clip. In general, soft real-time systems only guarantee that a deadline is met with a certain probability (obviously as near as possible to 100 percent). Motivated by the XML document statistics, we propose that a soft real-time system is sufficient enough to provide a high standard quality-of-service in SDI systems. Probability measures gained through the XML document statistics in combination with finite-state automatons in order to process XPath filtering expressions can be directly mapped onto the operating system level probability model.

6.2 DROPS Environment

For our prototypical implementation, we chose the Dresden Real-Time Operating System (DROPS, [11]) for our efforts spent in integrating OoS strategies into an SDI system. DROPS is based on the L4-micro-kernel family and aims to provide Quality-of-Service guarantees for any kind of application. The DROPS Streaming Interface (DSI, [17]) supports a time-triggered communication for producer-consumer-relationships of applications which can be leveraged for connecting data sources and data destinations to the filtering engine. The packet size of the transfer units (XML chunks) are variable and may therefore depend on the structure of the XML stream. Memory and CPU reservation is performed by a QoS resource manager. Since the management of computing time is based on the model of periodic processes, DROPS is an ideal platform for processing streaming data. A single periodic process reflects either an entire XML document at a macro level (as a unit of capacity planning) or single node of the XML document and therefore a single transition in an XPath filtering automaton at a micro level (as a unit of resource consumption, e.g. CPU usage, memory usage). Moreover schedulable and real-time capable components like a file system or a network connection exist to model data sources and destinations.

Figure 5 outlines the components performing the stream processing at the operating level. The query processor is connected through the DROPS Streaming Interface (DSI) to other components for implementing the data streaming

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

constraint by quality of service parameters given at the start of the filtering process at the macro level, i.e. for each XML document. The query engine is a streaming XPath processor based on the XML toolkit ([3]).

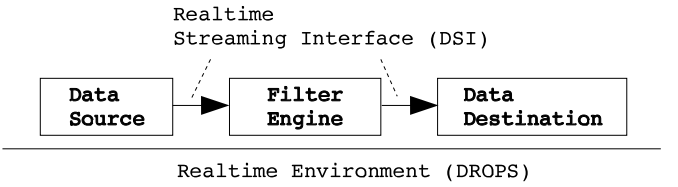

Fig. 5. connecting the involved components via the DROPS Streaming Interface

6.3 Adaption of the XML Toolkit

Due to the nature of SDI systems, stream based XPath processing techniques are more adequate for efficiently evaluating profiles against incoming XML documents because of the independence from document structure and document size. In our work we exploit the XML toolkit as a base for quality of service considerations. XMLTK implements XPath filtering on streaming data by constructing a deterministic finite automaton based on the registered XPath expressions. The prototypical implementation focuses on the core of XMLTK and ports the algorithms to the DROPS runtime environment (figure 6). Additionally the current implementation is extended to capture the following tasks:

- resource planning: Based on sample XML documents, the prototypical implementation plays the role of a proof-of-concept system with regard to the document statistics and the derivation of resource description at the operating system level.
- resource reservation: Based on the resource planning, the system performs resource reservation and is therefore able to decide whether to accept or reject a subscription with a specific quality-of-service requirement.
- filtering process scheduling: After the notification of an incoming XML document, the system provides the scheduling of the filtering process with the adequate parameters (especially memory and CPU reservation at the operating system level)
- monitoring filtering process: after scheduling according to the parameters, the system starts the filtering process, monitors the correct execution, and performs finalizing tasks like returning the allocated resources, etc.

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

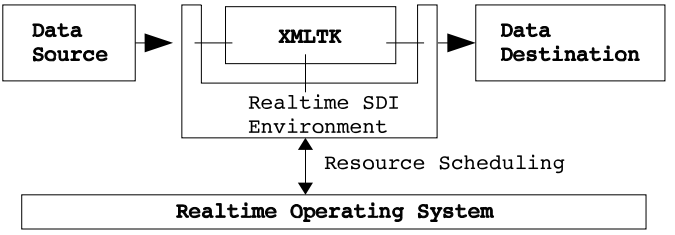

Fig. 6. XMLTK and the DROPS Environment

7 Summary and Conclusion

This paper introduces the concept of quality-of-service in the area of XML-based SDI systems. Currently discussed approaches in the SDI context focus on the efficiency of the filtering process but do not discuss detailed quality-of-service parameters. In this paper, we outline the motivation of Quality-of-Service in this application context, intensively discuss the two critical factors, XML document and XPath queries, to accurately estimate the resource consumption for a single XML document, and outline the requirements of the underlying real-time operating system. The current implementation is based on the DROPS operating system and extends the core components of the XML toolkit to parameterize the operating system. Although this paper sketches many points in the context of quality-of-service in XML-based subscription systems, we are fully aware that many issues are still open and therefore represent the subject of further research. However, filtering engines working on a best effort basis are definitely not the real answer to the challenge of a scalable and high-performing subscription system.

References

1. Altinel, M.; Aksoy, D.; Baby, T.; Franklin, M.; Shapiro, W.; Zdonik, S.: DBIS Toolkit: Adaptable Middleware for Large Scale Data Delivery in Proc. ACM SIGMOD Conference, Philadelphia, PA, pages 544–546, June 1999
2. Altinel, M.; Franklin, Michael J.: Efficient Filtering of XML Documents for Selective Dissemination of Information, in Proc. of the VLDB Conference, Cairo, Egypt, pages 53–64, September 2000
3. Avila-Campillo, I.; Green, T.J.; Gupta, A; Onizuka, M.; Raven, D.; Suciu, D. : XMLTK: An XML Toolkit for Scalable XML Stream Processing in Proc. of Programming Language Technologies for XML (PLAN-X) workshop, Pittsburgh, PA, October 2002
4. Chan, C.Y.; Felber, P.; Garofalakis, M.N.; Rastogi, R.: Efficient Filtering of XML Documents with XPath Expressions, in Proc. of the ICDE, San Jose, California, February 2002

Final edited form was published in "Database and XML Technologies. First International XML Database Symposium. Berlin 2003", S. 195-207, ISBN 978-3-540-39429-7
http://dx.doi.org/10.1007/978-3-540-39429-7_13

5. Chandrasekaran, S.; Cooper, O.; Deshpande, A.; Franklin, M.J.; Hellerstein, J.M.; Hong, W.; Krishnamurthy, S.; Madden, S.; Raman, V.; Reiss, F.; Shah, M.: TelegraphCQ: Continuous Dataflow Processing for an Uncertain World, in Proc. of the CIDR Conference, Asilomar, CA, January 2003
6. Chaudhri B. A., Rashid A., Zicari, R.: XML Data Management – Native XML and XML-Enabled Database Systems Addison-Wesley, 2003
7. Chen, J.; DeWitt, David J.; Tian, F.; Wang, Y: NiagaraCQ: A Scalable Continuous Query System for Internet Databases, in Proc. of the: ACM SIGMOD Conference on Management of Data, Dallas, Texas, pages 379–390, May 2000
8. Diao, Y.; Fischer, P.; Franklin, Michael J.; To, R.: YFilter: Efficient and Scalable Filtering of XML Documents, in Proc. of the ICDE Conference, San Jose, California, pages 341–342, February 2002
9. Green, T.J.; Miklau, G.; Onizuka, M.; Suciu, D.: Processing XML Streams with Deterministic Automata, in Proc. of ICDT, Siena, Italy, pages 173–189, January 2003
10. Hamann, C.-J.; Märcz, A.; Meyer-Wegener, K.: Buffer Optimization in Realtime Media Servers using Jitter-contrained Periodic Streams, technical report, TU-Dresden, January 2001
11. Härtig, H.; Baumgartl, R.; Borris, M.; Hamann, C.-J.; Hohmuth, M.; Mehnert, F.; Reuther, L.; Schönberg, S.; Wolter, J.: DROPS - OS Support for Distributed Multimedia Applications, in Proc. of the ACM SIGOPS European Workshop, Sintra, Portugal, September 7–10, 1998
12. Ives, Zachary G.; Halevy, Alon Y.; Weld, S. Daniel: An XML Query Engine for Network-Bound Data, in: VLDB Journal 11(4), pages 380–402, 2002
13. Klettke, M., Meyer, H.: XML & Datenbanken dpunkt.verlag, 2003
14. Lehner, W.: Subskriptionssysteme – Marktplatz für omnipräsente Informationen, Teubner Texte zur Informatik, Band 36, B.G. Teubner Verlag Stuttgart/Leipzig/Wiesbaden, 2002
15. Lehner, W.: Datenbanktechnologie für Data-Warehouse-Systeme, dpunkt.verlag, Heidelberg, 2003
16. Lehner, W.; Irmert, F.: XPath-Aware Chunking of XML Documents, in Proc. of GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW) Leipzig, Germany, pages 108–126, February 2003
17. Löser, J.; Härtig, H.; Reuther, L.: A Streaming Interface for Real-Time Interprocess Communication, technical report, TU-Dresden, August 2001
18. Ludäscher, B.; Mukhopadhyay, P.; Papakonstantinou, Y.: A Transducer-Based XML Query Processor, in Proc. of the VLDB Conference, Hongkong, China, pages 227–238, August 2002
19. Mannino, M. V., Chu, P., Sager, T.: Statistical Profile Estimation in Database Systems in: ACM Computing Surveys, 20(3), 1988, pages 191–221